


Suffix Structures

Blue 11 - Lecture 01: Dynamic Array & String

Tóm tắt đề bài

Tóm tắt đề bài

- Cho 2 chuỗi phân biệt **s** và **t**.
- Hãy chuyển chuỗi **s** thành chuỗi **t** thông qua các phép biến đổi:
 1. **Automaton**: Xóa 1 ký tự bất kỳ trong chuỗi **s**
 2. **Array**: Hoán đổi vị trí của hai ký tự bất kỳ trong chuỗi **s**
- Xác định:
 - **s**  **t**?
 - Nếu chuyển được thì sử dụng phép biến đổi nào ?
 - Biết rằng số lần biến đổi là không giới hạn

Mô tả Input/Output

Input

- $|s| > 0$
- $|t| > 0$
- $s \neq t$
- s và t chỉ gồm chữ cái latin in thường

Output

- “**need tree**” nếu không thể biến s thành t
- “**automaton**” chuyển s thành t được thông qua phép biến đổi số 1
- “**array**” chuyển s thành t được thông qua phép biến đổi số 2
- “**both**” nếu cần thực hiện cả 2 phép biến đổi

Giải thích ví dụ

Ví dụ 1

Input

```
s = 'automaton'  
t = 'tomat'
```

Output

```
automaton
```

Giải thích:

```
s = 'automaton'  
t = 'tomat'
```

Ví dụ 2

Input

```
s = 'array'  
t = 'arary'
```

Output

```
array
```

Giải thích:

```
s = 'aray' → swap r với a ta được s = 'aray'  
t = 'arary'
```

Ví dụ 3

Input

s = 'both'

t = 'hot'

Output

both

Giải thích:

s = '~~b~~oth' → **xóa b** ta được s = 'oth'

→ **swap** (o,h) thì s = 'hto'

→ **swap** (t, o) và s =

'hot' = t

Ví dụ 4

Input

s = 'need'

t = 'tree'

Output

need tree

Giải thích: để biến s thành t thì trong s phải có ít nhất 1 chữ r và 1 chữ t → không có cách nào biến đổi s thành t được.

Hướng dẫn giải

Nhận xét

- TH1: Nếu 1 kí tự nào đó có trong t mà không có trong s → **'need tree'**
- TH2: Nếu 1 kí tự có trong s mà không có trong t thì xóa kí tự đó đi → **'automaton'**
- TH3: Nếu thứ tự xuất hiện của các ký tự trong t không khớp với s → **'array'**
- TH4: Kết hợp TH2 + TH3 → **'both'**

Cách giải

Để giải quyết trường hợp **TH1** (**'need tree'**) và **TH2** (**'automaton'**):

- Tạo mảng đếm tần suất của các kí tự trong chuỗi s và t: `cnt_s` và `cnt_t`
- Đối với từng kí tự x trong bảng chữ cái latin (26 chữ cái):
 - Nếu `cnt_t[x] > cnt_s[x]` → **'need tree'**
 - Ngược lại: `cnt_t[x] < cnt_s[x]` → **'automaton'**

Cách giải

Input

s = 'automaton'

t = 'tomat'

Output

automaton

Minh họa mảng cnt_s và cnt_t cho trường hợp 1 và 2:

	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z
index	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25
cnt_s	2												1	1	2					2	1					

	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z
index	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25
cnt_t	1												1		1					2						

Cách giải

- Để giải quyết trường hợp **TH3 'array'**:
 - Gọi match là vị trí xuất hiện của ký tự trước đó của t trong s
 - Duyệt qua từng ký tự t[i] trong chuỗi t
 - Tìm vị trí xuất hiện đầu tiên của t[i] trong s mà xuất hiện sau match
 - Nếu các vị trí này tăng dần nghĩa là các ký tự này đã đúng vị trí, ta không cần đổi chỗ. Ngược lại ta kết luận "array"
- Trường hợp **TH4 'both'**: nếu cần cả '**automaton**' và '**array**'

Độ phức tạp: $O(\max(\text{length}(s), \text{length}(t)))$

Cách giải

Input

s = 'array'

t = 'arary'

Output

array

Minh họa trường hợp 3 (cần array)

i	t[i]	id t in s (find first occurrence of t[i] in s from match+1)	match	s	array
			-1	array	False
0	a	0	0	a rray	False
1	r	1	1	ar rray	False
2	a	3	3	arra y	False
3	r				True
4	y				

Cách giải

Input

s =
'automaton'

t = 'tomat'

Output

automaton

Minh họa trường hợp 3 (không cần array)

i	t[i]	id t in s (find first occurrence of t[i] in s from match+1)	match	s	array
			-1	automaton	False
0	t	2	2	au tomaton	False
1	o	3	3	auto maton	False
2	m	4	4	autom aton	False
3	a	5	5	automa ton	False
4	t	6	6	automato n	False

Mã giả

Mã giả

```
#input
read s, t
cnt_s, cnt_t = [0] * 26
for char in s:
    cnt_s[c] += 1
for char in t:
    cnt_t[c] += 1
automaton = array = need_tree = False
```

```
#case 1,2
for i := 0 to 25 do:
    if cnt_s[i] > cnt_t[i]:
        automaton = True
    if cnt_s[i] < cnt_t[i]:
        need_tree = True
```

Mã giả

```
#case 3
match = -1
for i := 0 to len(t) - 1 do:
    id = find first of t[i] in s after match
    if id != -1:
        match = id
    else: # id == -1
        array = True
```

```
#print result
if need_tree:
    print("need_tree")
elif automaton and array:
    print("both")
elif automaton:
    print("automaton")
else:
    print("array")
```

Thank you